


ORIGINAL RESEARCH

An interpretable semi-supervised system for detecting cyberattacks using anomaly detection in industrial scenarios

Ángel Luis Perales Gómez¹  | Lorenzo Fernández Maimó¹ | Alberto Huertas Celdrán² | Félix J. García Clemente¹

¹Departamento de Ingeniería y Tecnología de Computadores, University of Murcia, Espinardo, Murcia, Spain

²Communication Systems Group CSG, Department of Informatics IfI, University of Zurich, Zurich, Switzerland

Correspondence

Ángel Luis Perales Gómez, Departamento de Ingeniería y Tecnología de Computadores, University of Murcia, Espinardo, Murcia, Spain.
Email: angelluis.perales@um.es

Funding information

MCIN/AEI/10.13039/501100011033/FEDER, UE, Grant/Award Number: PID2021-122466OB-I00; Swiss Federal Office for Defence Procurement, Grant/Award Number: Aramis R-3210/047-31; MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, UE, Grant/Award Number: TED2021-129300B-I00

Abstract

When detecting cyberattacks in Industrial settings, it is not sufficient to determine whether the system is suffering a cyberattack. It is also fundamental to explain why the system is under a cyberattack and which are the assets affected. In this context, the Anomaly Detection based on Machine Learning (ML) and Deep Learning (DL) techniques showed great performance when detecting cyberattacks in industrial scenarios. However, two main limitations hinder using them in a real environment. Firstly, most solutions are trained using a supervised approach, which is impractical in the real industrial world. Secondly, the use of black-box ML and DL techniques makes it impossible to interpret the decision made by the model. This article proposes an interpretable and semi-supervised system to detect cyberattacks in Industrial settings. Besides, our proposal was validated using data collected from the Tennessee Eastman Process. To the best of our knowledge, this system is the only one that offers interpretability together with a semi-supervised approach in an industrial setting. Our system discriminates between causes and effects of anomalies and also achieved the best performance for 11 types of anomalies out of 20 with an overall recall of 0.9577, a precision of 0.9977, and a F1-score of 0.9711.

KEYWORDS

anomaly detection, deep learning, explainable artificial intelligence, industry applications, machine learning, root cause analysis

1 | INTRODUCTION

The industry is moving towards the Industry 4.0 paradigm, whose fundamental characteristic consists of automating industrial processes by introducing interconnected smart devices in factories. These smart devices provide new and exciting features in the industrial scenario related to how humans and devices interact. Some examples could be the remote control of devices or a sensorized factory that increases workers' safety. To ease such features, typical devices introduced in industrial scenarios are actuators and sensors that generate time-series data with cause-effect relationships. In addition to new hardware equipment, Industry 4.0 is also about introducing new technologies to manage the behaviour of that hardware and

extract information from the data gathered by the smart devices. Some of these technologies are Big Data [1] and Artificial Intelligence [2].

However, these new features came at a price. Factories have been isolated from external networks for decades, but now they are connected to the Internet to bring all these features [3]. As a result, the number of cyberattacks affecting industrial factories has increased in the last years [4]. To protect the factories, the research community has adopted the Anomaly Detection (AD) paradigm to detect highly specialised cyberattacks affecting the industry. These cyberattacks are different from those launched in traditional data networks. In this last scenario, cyberattackers take advantage of software vulnerabilities to launch well-known cyberattacks such as

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial](https://creativecommons.org/licenses/by-nc/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2023 The Authors. *IET Information Security* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

Denial of Service (DoS) or SQL Injection. However, in industrial scenarios, the main goal is to modify the process behaviour to produce a competitive and economical loss [5]. The AD paradigm is based on modelling the normal behaviour system to draw the boundary of normality. All behaviour outside of this boundary is considered a cyberattack. In practice, one of the most effective ways to draw that boundary is through Machine Learning (ML) and Deep Learning (DL) techniques.

However, AD systems using ML/DL present two main limitations that hinder their wide implementation in the real industrial world. On the one hand, the lack of interpretability and/or explainability of AD systems based on ML and DL techniques makes it difficult to understand the decisions made by these systems. Most AD systems in the literature inform whether an anomaly is detected but not what sensor or actuator is affected by the anomaly. By providing this information, operators and administrators can dedicate their efforts to the specific point responsible of the anomaly. Therefore, interpretable AD systems save resources when selecting the proper mitigation actions. On the other hand, a vast number of ML and DL models used in industrial AD systems are trained using a supervised approach because it tends to achieve the best results. This means that a fully labelled dataset is required to train these systems. However, this type of dataset is not always available, and their generation requires investing a significant amount of resources [5], being, therefore, impractical in real scenarios. On these grounds, a semi-supervised approach that enables training AD systems using a dataset labelled only with normal samples reduces the effort needed to train such systems in industrial scenarios.

To overcome the previous challenges, this work presents the following contributions:

- The relationship between the four desirable properties of interpretable ML/DL models and the industrial scenarios. In particular, for each property, we discuss the expected outcome.
- A system specially designed for industrial scenarios composed of six steps to train semi-supervised and interpretable models. To discriminate between normal and abnormal samples, the system computes a threshold based on statistic measures. When a sample exceeds such threshold, it is categorised as abnormal. This system pays particular attention to the interpretability step where causal inference model are used to discriminate between causes and effects of anomalies. These techniques limit the usage of our system to environments whose devices generate time-series data with cause-effect relationships such as industrial scenarios. Besides, the system presents specific steps focussed on the feature extraction and AD method to remove both highly correlated features and potential outliers during threshold computation.
- The validation of the system in a realistic industrial scenario called Tennessee Eastman Process (TEP) [6]. We proved that our proposal not only discriminated between causes and effects but also it achieved the best performance detection

for 11 types of anomalies out of 20 with an overall recall of 0.9577.

The remainder of this paper is structured as follows. Section 2 discusses the different works focussed on AD in industrial scenarios. Section 3 describes the desirable properties of interpretable models in industrial scenarios. In Section 4, the interpretable and semi-supervised system is introduced. Section 5 presents the system validation using the dataset collected from the TEP. Finally, Section 6 presents the conclusions and the future work.

2 | RELATED WORK

This section discusses the works in the literature related to AD in industrial scenarios. Besides, we review the existing methodologies to detect anomalies in industrial scenarios.

Anomaly Detection is a vast area of study, and different methods have been proposed [7]. However, due to the highly specialised cyberattacks affecting the industry, the most successful probed techniques are based on ML and DL. In this context, the authors of ref. [8] proposed a non-interpretable and unsupervised approach based on a Variational Long Short-Term Memory (VLSTM) that tries to reconstruct the input features. Besides, the authors considered three loss functions to constrain the hidden variable during the learning phase. To validate its proposal, the authors utilised a public dataset named UNSW-NB15, and they probed that VLSTM improves accuracy and reduces the false rate. The same authors highlighted the difficulty of getting labelled datasets, and they proposed another non-interpretable approach based on few-shot learning and Siamese Convolutional Neural Networks (FSL-SCNN) [9]. Similar to the previous work, the authors proposed three loss functions, and the same dataset was used. Regarding the performance, this work achieved good results, although lower than the VLSTM approach.

The authors of ref. [10] presented a novel hybrid deep random neural network (HDRaNN) for AD, with a non-interpretable and supervised approach. In particular, the approach combined a deep random neural network and a multilayer perceptron (MLP) with dropout regularisation to avoid overfitting. The network comprised an input and output layer, three recurrent layers, and three MLP. The authors tested the architecture with two different datasets: DS2OS and UNSW-NB15. The authors of ref. [11] proposed another non-interpretable and unsupervised intrusion detection system based on Stacked AutoEncoders (SAE) to distinguish between normal and abnormal behaviour in real-time. In particular, the SAE proposed is based on LSTM neural networks, and it was validated using the gas pipeline and UNSW-NB15 datasets, where the solution achieved an F1-score of 97.89% and 97.55% respectively. In terms of ensembles, the authors of ref. [12] presented a novel architecture that combined multiple non-interpretable and supervised DL and ML models to detect cyberattacks in industrial scenarios. In particular, they proposed splitting the original dataset into various balanced datasets

passed to diverse SAE to learn different data representations. Then, the new data transformed by SAE were passed to different Fully Connected Neural Networks (FCNN). Their outputs were combined to form a super vector that finally was passed to a Decision Tree (DT) that performed the classification. This approach showed to be efficient when dealing with an imbalanced dataset. In particular, the authors tested the approach using a Gas pipeline and SWaT dataset, improving the state-of-the-art results. The authors of ref. [13] presented an unsupervised and non-interpretable approach that combines the usage of an FCNN and a One-Class Support Vector Machine (OC-SVM). Besides, the authors presented a new regularisation term that provides a model-tuning mechanism based on specific industrial requirements and performance metrics of interest. The approach was tested on the SWaT dataset and achieved the state-of-the-art result in 15 out of 36 attacks.

Because of the high dimensional data present in industrial scenarios, the authors of ref. [14] proposed a non-interpretable and supervised approach based on Generative Adversarial Network (GAN) and Deep Belief Network (DBN) together with some techniques to remove highly correlated features. Finally, the authors of ref. [15] showed that a modification of Principal Component Analysis (PCA) performed well in industrial scenarios. In particular, the authors proposed the usage of Multivariate Generalized Likelihood Ratio (GLR) and Moving Window Interval Aggregation (MWIA) together with Interval PCA (IPCA), which is a non-interpretable and semi-supervised approach. The authors showed that the proposed approach achieved state of the art in terms of general performance.

Regarding solutions that adopt interpretability or explainability methods, we highlight [16], where the authors designed a supervised approach based on Layer-wise Relevance Propagation (LRP) and FCNN. Using those techniques, an intrusion detection model can be trained with a training dataset that does not contain too much information. Such intrusion detection model also have the ability to quickly analyse the relevance of each feature. The authors validated their solution by using the Gas Pipeline dataset. About solutions focussed on the analysis of root causes, we highlight two of them. On the one hand, the authors of ref. [17] applied the Joint Recurrence Plot (JRP) and the Density-based Spatial Clustering Application with Noise (DBSCAN) algorithm. On the other hand, the authors of ref. [18] presented a fault diagnosis framework. In particular, they computed the Mutual Information (MI) between each pair of features in the training dataset. When an anomaly was detected, each pair of features with MI beyond the one computed in the training dataset is examined by means of Time Delayed Mutual Information (TDMI) analysis to determine the causal logic between them.

In terms of algorithm comparison, the authors of ref. [19] presented an analytical study of AD in industrial scenarios. The dataset used was extracted from a Supervisory Control And Data Acquisition (SCADA) that controls an aquatic storage and distribution system. In particular, the authors evaluated six ML techniques: Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbours (KNN), Classification And Regression Tree (CART), Gaussian Naive Bayes (NB), and

Support Vector Machine (SVM). In terms of performance, CART was the model that achieved the best F1-score (86%). Another work that studied the performance of different ML and DL models in industrial scenarios was presented in ref. [20]. In this case, the authors tested LR, Lasso, SVM, DT, Adaptive Boosting (AB), Gradient Boosting (GB), Random Forest (RF), and FCNN. The datasets selected to test these models were the Gas pipeline and the TEP datasets. As shown by the authors, the FCNN and RF achieved the best performance. The authors of ref. [21] presented another study on how the number of hidden layers, number of neurons in the last hidden layer, and data augmentation impact the detection performance. The authors tested their solution in fault detection and classification setting using the TEP dataset. With respect to the number of layers, the authors showed that using two to four hidden layers achieved the best performance, being the difference between them insignificant. In terms of the number of neurons in the last hidden layer, the authors showed that having more neurons on that layer improved the detection performance. Another study of ensemble methods can be found in ref. [22]. In this study, the authors tested supervised approaches that offer interpretability mechanisms such as DT, RF, GB, AB, Light Gradient Boosting (LGBM), Extreme Gradient Boosting (XGBoost), and CatBoost (CB). To deal with the imbalance problem of AD problems, the authors applied the Synthetic Minority Over-sampling Technique (SMOTE). Furthermore, the authors combined the previous ML models together with more powerful interpretability mechanisms such as LIME and SHAP. Finally, the authors concluded that XGBoost reached the best performance when they tested on a heating, ventilation, and air conditioning (HVAC) dataset [23]. In ref. [5], another comparison of different ML and DL models were presented. The models were evaluated using Electra, a dataset collected from an Electric Traction Substation. The authors studied the performance of RF, SVM, OC-SVM, Isolation Forest (IF), and FCNN, concluding that the models that achieved the highest performance was SVM and RF. Besides, the same authors presented another study comparing the performance of these models in terms of time evaluation [24]. The conclusion was that FCNN outperforms the other models.

Respecting methodologies available in the literature, we highlight MADICS [25] and the one presented in ref. [26]. Both are semi-supervised methodologies. However, the first one is oriented to training AD systems from normal behaviour, while the second is focussed on training AD using a subset of normal and abnormal behaviour. MADICS is a complete methodology because it recommends specific actions for each step that must be carried out in an AD task in industrial scenarios. However, the methodology presented in ref. [26] only proposes a model with no recommendations about steps like feature filtering or feature extraction. Although MADICS is a complete methodology, it does not consider the potential outliers when computing the threshold. Moreover, the features extracted by MADICS were highly correlated because they were computed using aggregate functions like minimum or range. Finally, although MADICS gives a few recommendations about interpretability, the methodology does not cover

this point completely, and it was not tested in this specific aspect.

Table 1 shows a comparison between all the solutions described above. As can be seen, a few of the proposed solutions in the literature are focussed on semi-supervised and interpretable approaches. In fact, most of the solutions that offer interpretability are based on ML, and the interpretability method relies on such an ML model. This work presents a solution that overcomes the MADICS limitation commented above. To the best of our knowledge, this solution is the only one that combines interpretability and semi-supervised approach in the industrial setting. First, we extract higher-order features using techniques suitable in industrial scenarios. Second, we remove the outliers when computing the threshold, preventing them from impacting the threshold computed. Third, we introduce a new step to achieve interpretability, allowing operators to identify the exact point where anomalies are produced.

3 | DESIRABLE PROPERTIES OF INTERPRETABLE ML/DL IN INDUSTRIAL SCENARIOS

This section introduces the interpretability in ML/DL models together with the properties of interpretability methods and their relationship to industrial scenarios.

Interpretability is the degree to which a human can understand the cause of a decision [27]. In many scenarios, this property becomes mandatory since it allows us to obtain very valuable information. For example, imagine an AD system in an industrial setting that does not allow us to interpret the result of a specific prediction. In this case, when a cyberattack happens, we can detect it but we cannot know which specific parts of the industrial system are affected. In contrast, if the AD system deployed is based on an interpretable model or offers a specific interpretable mechanism, we can know which specific part is being attacked and make concrete decisions to mitigate the cyberattack. In this way, interpretable models allow us to take more effective actions at a lower cost when mitigating cyberattacks.

However, interpretability is not always easy to achieve when using ML/DL models. In most cases, these techniques are not interpretable out of the box, and it is necessary to resort to other methods to interpret the results. In particular, within the most popular ML models, RF, Linear Regression, and LR offer certain degrees of interpretability. Unfortunately, DL models have no built-in mechanism to interpret the prediction results. For both DL and ML models that do not offer this feature, a wide variety of third-party tools are available that allow a certain degree of interpretability.

The authors of ref. [28] identified four properties of an interpretability method that can be used to compare different

TABLE 1 Comparison of different AD solutions focussed on industrial scenarios.

Sol.	Validated on	Model	Interpretability	Approach
[8]	UNSW-NB15	VLSTM	✗	Unsupervised
[9]	UNSW-NB15	FSL-SCNN	✗	Supervised
[10]	DS2OS, UNSW-NB15	HDRaNN	✗	Supervised
[11]	Gas pipeline, UNSW-NB15	SAE	✗	Unsupervised
[12]	SWaT, gas pipeline	Ensemble of SAE, FCNN, and DT	✗	Supervised
[13]	SWaT	FCNN and OC-SVM	✗	Unsupervised
[14]	TEP	DBN and GAN	✗	Supervised
[15]	TEP	IPCA	✗	Semi-supervised
[16]	Gas pipeline	FCNN	✓	Supervised
[19]	Aquatic storage and distribution dataset	LR, LDA, k-NN, CART, NB, and SVM	- ^a	Supervised
[20]	TEP, gas pipeline	LR, Lasso, SVM, DT, AB, GB, RD, and FCNN	- ^a	Supervised
[21]	TEP	FCNN	✗	Supervised
[22]	HVAC dataset	DT, RF, GB, AB, LGBM, XGBoost, CB	✓	Supervised
[5]	Electra	RF, SVM, FCNN, OC-SVM, and IF	- ^a	Supervised, semi-supervised
[26]	TEP	LSTM	✗	Semi-supervised
[25]	SWaT	LSTM	✓	Semi-supervised
Ours	TEP	LSTM	✓	Semi-supervised

Abbreviations: AB, Adaptive Boosting; DBN, Deep Belief Network; DT, Decision Tree; FCNN, Fully Connected Neural Networks; FSL-SCNN, few-shot learning and Siamese Convolutional Neural Networks; GAN, Generative Adversarial Network; GB, Gradient Boosting; HDRaNN, hybrid deep random neural network; HVAC, heating, ventilation, and air conditioning; IPCA, Interval PCA; LGBM, Light Gradient Boosting; RF, Random Forest; SAE, Stacked AutoEncoders; TEP, Tennessee Eastman Process; VLSTM, Variational Long Short-Term Memory.

^aSome of the models proposed offer interpretability properties.

solutions. These properties are described below, along with their relationship to industrial scenarios.

- **Expressive Power.** It refers to the mechanism by which the decisions reached by the models are interpreted. Examples of these mechanisms are languages based on IF-THEN, decision trees, thresholds, or graphs. Concerning industrial environments, it is desirable that the mechanism selected is easily and quickly understood by operators to allow fast detection of the sensor/actuator affected when a cyber-attack happens.
- **Translucency.** This property tells us the relationship between the interpretability mechanism and the model and its parameters. For example, models with inherent interpretability mechanisms, such as RF, offer a high degree of translucency. In general, methods with a high degree of translucency have access to more information and, therefore, allow the interpretation of the result out of the box. However, these mechanisms depend on the specific ML/DL model that implements them, and, in critical fields such as AD in industrial environments, it is possible to change the model when another one with better performance is obtained.
- **Portability.** This property refers to the range of ML/DL models applicable to the interpretability mechanism. This property is closely related to the previous one because highly portable methods offer a low degree of translucency and vice versa. In relation to industrial environments, highly portable mechanisms are preferred because they can be used with a wide range of models.
- **Algorithmic Complexity.** This property describes the algorithmic complexity of the interpretability mechanism. In industrial settings, the complexity needs to be low in order to carry out detection and interpretation as quickly as possible.

In Table 2, the previously discussed properties together with the expected option in the industrial scenarios are showed.

4 | A SEMI-SUPERVISED AND INTERPRETABLE APPROACH TO ANOMALY DETECTION IN INDUSTRIAL SCENARIOS

This work presents an interpretable and semi-supervised system composed of the typical steps in ML/DL.

1. **Data preprocessing.** In general, the first task usually carried out in this step is to split the dataset into training, validation, and test datasets. The second task is in charge of studying all features to discover spurious or corrupted values. Finally, it is recommended to perform categorical data encoding and scale features.
2. **Feature filtering.** This step applies different techniques to remove features that do not provide useful information to the model. To this task, we can combine three approaches. The first one is to study the correlation between the features and the label to determine if data leakage exists in the dataset. The second one is to study the variance of feature values and remove those that do not change in the whole dataset. Finally, the third approach determines if the statistical distribution is preserved in the training, validation, and test datasets, removing those features whose distribution is not preserved.
3. **Feature extraction.** This step consists in generating higher-order features that discriminate between normal and abnormal system behaviour. Due to the repetitive nature of time-series data generated in industrial scenarios, it is recommended the usage of techniques that can deal with such data and extract patterns from them.
4. **Anomaly Detection method.** In general, this step is in charge of training and fine-tuning the AD model. First, an ML/DL model must be selected. In industrial scenarios, and considering the time-series data generated in such scenarios, good options are the CNN or LSTM models. On the one hand, these algorithms can model data patterns of higher complexity than the ML approach. On the other hand, LSTM and CNN models handle time-series data out of the box, which is not possible using simpler ML models. Then, a typical second task is to select the range of the hyperparameters used to train and fine-tune the model. Finally, when using a semi-supervised approach based on a regressor model, it is also required to compute a threshold or set of thresholds from which a sample will be classified as abnormal. In general, the z-score offers good properties to compute such thresholds.
5. **Validation.** In this step, the model trained in the previous step is validated. To this end, proper metrics need to be chosen. In particular, precision, recall, and F1-score are widely used metrics in AD tasks. Finally, it is required to analyse the obtained result to check if they are appropriate.

Despite the previous steps, the core of our proposal is the introduction of the interpretability step, together with the

TABLE 2 Desirable properties of interpretable mechanisms and their expected option in industrial scenarios.

Property	Description	What is expected in industrial scenarios?
Expressive power	Refers to the mechanism by which the decisions reached by the models are interpreted	Easy to understand
Translucency	Relationship between the interpretability mechanism and the model and its parameters	Low translucency
Portability	ML/DL models range applicable to the interpretability mechanism	High portability
Algorithmic complexity	Time required to apply the interpretability mechanism	Low algorithmic complexity

contribution to the Feature Extraction and Anomaly Detection Method steps. In particular, the new interpretability step combines the previously extracted features and the computed threshold to figure out the potential causes of the anomalies. Then, different causal relationship discovery techniques are applied to discriminate between effects and causes. This step limits our solution to environments whose devices generate time-series data with cause-effect relationship. Regarding the contribution to the Extract Features step, we apply autocorrelation and Discrete Fourier Transform (DFT) techniques and extract the dominant coefficients. Besides, we propose a technique to filter outliers when computing the error between the ground truth and the values predicted by the ML/DL model. Without such outliers, we can compute a more robust threshold to determine whether a specific sample is normal or abnormal.

In the following subsections, we detail the modification proposed to the Feature Extraction and Anomaly Detection Method. Furthermore, we also introduce the proposed approach to determine the root cause of the anomaly.

4.1 | Feature extraction

Generally, in their task of controlling and monitoring equipment, industrial devices perform repetitive actions. Therefore, the feature extraction is carried out by means of two techniques that exploit that behaviour. First, we apply the autocorrelation function to a window so as to extract higher-order features. The formal definition of autocorrelation is shown in Equation (1).

$$autocorr_{x_w,k} = \frac{\sum_{i=w-W+1}^{w-k} (x_i - \hat{x})(x_{i+k} - \hat{x})}{\sum_{i=w-W+1}^w (x_i - \hat{x})^2} \quad (1)$$

where k is the lag introduced, W is the length of the window, x_w is the value of x at instant w , and \hat{x} is the mean of the x values in the window.

Then, we use the DFT applied to a window to extract higher-order features. In particular, DFT provides an easy way to convert a signal from time to frequency domain. The formal definition of DFT is shown in Equation (2).

$$\overline{DFT}_{x_w,k} = \sum_{j=w-W+1}^w x_j e^{-\frac{2\pi i}{W} k(j-(w-W+1))} \quad (2)$$

where x_w is the value of x at instant w and W is the total number of samples.

Once those two techniques are applied to a window, we suggest an approach based on considering the dominant values of the output of each technique. In particular, the dominant values of the autocorrelation technique give an idea of how similar is the time-series signal with the same signal in the future. Regarding the DFT technique, the dominant values or frequencies correspond to frequencies with the highest

amplitudes in the signal. Assume that an anomaly originated by a cyberattack in industrial scenarios will cause an alteration in these values.

4.2 | Anomaly detection method

This step is in charge of selecting the proper model, its hyperparameters, training and fine-tuning the models, and selecting the threshold to detect anomalies. In particular, we propose to select the threshold based on the z-score of prediction error. First, the error of each sample between the ground truth and the predicted result in the training dataset is computed. Then, the mean (μ_e) and standard deviation (σ_e) of these errors are computed. Next, the error for each sample in the validation dataset is computed, and finally, to normalise the errors, the mean (μ_e) is subtracted from the validation prediction errors, e_v , and it is divided by the standard deviation (σ_e) as shown in Equation (3).

$$z_e = \frac{e_v - \mu_e}{\sigma_e} \quad (3)$$

Based on this metric, we propose to select the largest z-score as the threshold. This threshold will serve to discriminate between normal and abnormal samples. However, the main drawback of this approach is that it includes the error outliers in the computation of the highest z-score value. To overcome this limitation, we propose using the Inter-Quartile Range (IQR), which is a more robust metric in the presence of outliers [29]. The IQR is computed by subtracting the first quartile of the z-score, q_1 , from the third quartile of the z-score, q_3 as shown in Equation (4).

$$IQR = q_3 - q_1 \quad (4)$$

Since we are not concerned about potential outliers with very small values, we compute the threshold as a function of IQR and the third quartile of the previously computed z_e as shown in Equation (5). This method helps to ignore outliers present in the data during the threshold selection.

$$threshold = q_3 + 1.5 * IQR \quad (5)$$

4.3 | Interpretability

Here, some of the previous steps are taken into account to make the trained model interpretable. The proposed interpretability method is based on figuring out which is the specific sensor/actuator that is affected by the anomaly. For example, suppose that a cyberattack aims to alter the behaviour of a reactor. In that case, this method will inform us not only that the cyberattack is taking place but also that it is also affecting the reactor. In this way, the plant operators will be able to

invest time and effort in mitigating the cyberattack by putting all their efforts into the reactor.

As explained before, an anomaly is detected when the deviation between the actual and predicted values of any of the feature exceeds a certain threshold. In this case, that deviation would also contribute to interpret what is the cause of the anomaly. However, the fact of using a feature extraction scheme prevents us from using a naive approach and, therefore, forces us to use a procedure where the different previously calculated features are combined.

To solve the aforementioned problem, we propose grouping features in different sets depending on the original feature from which they were derived. In other words, if we have a specific feature, x , we consider it together with the autocorrelation features obtained from this feature, $x_{autocorr}$, and the frequency features obtained from it, x_{DFT} . In this way, when interpreting which is the sensor/actuator affected by the anomaly, our system will report the sensors/actuators where at least one of their related features exceed the threshold. After carrying out the AD Method step, the features that exceed the threshold will be considered as the affected ones.

However, we need to figure out the relationship between these affected features to determine the original cause of the anomaly. Therefore, we propose to build a graph that represents the relationship between those features. To build the adjacency weighted matrix representing the graph, we use the Distance Correlation (dCor) for three reasons. The first one is its suitability to detect novel and non-linear relationships [30]. The second is the lower asymptotic time complexity of dCor versus other techniques. Specifically, dCor is $\mathcal{O}(n^2)$ [31], being n the number of samples, whereas the time complexity of Maximal Information Coefficient (MIC) is $\mathcal{O}(n^{2.4})$ [32] and the time complexity for Heller-Heller-Gorfine (HHG) is $\mathcal{O}(n^2 \log n)$ [33]. The third reason is that dCor is not only a dependence test; it also returns the linear and non-linear association between two features, allowing us to establish the relationship between them in a scenario under attack. Nonetheless, the constructed graph presents two drawbacks. The first one is that the graph will be fully connected, since all the similarities will be greater than 0. Additionally, it will also represent the underlying similarity of the normal behaviour, in which we are not interested. The second drawback is that the resulting graph is undirected, and, therefore, we cannot distinguish which feature is the root cause and which features are the effects.

To solve the first problem, we suggest computing the dCor values between each pair of features for the normal samples contained in the validation dataset, using these values as additional thresholds. Concretely, dCor is a measure of dependence between two vectors that do not necessarily have the same dimension. The idea is to calculate the dCor for each pair of features that exceeded the threshold during the previous step in each anomaly. However, this measure also includes any intrinsic relationships associated with normal behaviour. Therefore, we propose to determine the increase of similarity between the scenario under attack and the normal scenario. In this way, if the dCor value of two features from abnormal

samples is lower than the dCor value obtained for these features in the validation dataset, that only contains normal samples, the value of the adjacency matrix in that position will be 0, removing the link between those features. This approach tries to relate features based on the similarity in the abnormal behaviour, ignoring its relationship in normal functioning.

To solve the second problem, we suggest using the well-known technique called Information-Geometric Causal Inference (IGCI) [34] that, given two features, determines if they are dependent or not, and in the case that they are dependent, IGCI indicates which is the cause and which is the effect. Information-Geometric Causal Inference is a pairwise causal discovery model considering the case of minimal noise $Y = f(X)$ with f invertible and leverages asymmetries to predict causal directions.

Table 3 summarises the properties offered by our interpretability mechanism. To be specific, the mechanism offers an expressive power based on a relationship graph that is intuitive and easily understandable by operators. Besides, our method also provides low translucency and high portability, which are desirable properties in an industrial scenario. Additionally, this interpretability method is model-independent. Finally, its algorithmic complexity is $\mathcal{O}(n^2 m)$, being m the number of features that exceeds the threshold. This algorithmic complexity arises from the dCor function. Since only a certain number of features will be considered, and it is not necessary to perform the process in real time, it is suitable for industrial environments.

5 | EXPERIMENTAL RESULTS

This section presents the TEP scenario and the dataset generated from it, detailing all the steps carried out to train our AD model. In order to perform the experiment, we used the following tools: Keras [35] to train our DL model, Scipy [36] to determine if samples came from the same distribution and to compute correlation and variance study, Causal Discovery Toolbox [37] to compute the IGCI, and Pingouin [38] to calculate the dCor values.

5.1 | Dataset

In this work, we validate our proposal using the TEP [6], which is a chemical simulation testbed widely used in AD. This testbed introduces anomalies related to faults in the industrial process.

TABLE 3 Properties offered by our interpretability method.

Property	Our method
Expressive power	Graph
Translucency	Low
Portability	High
Algorithmic complexity	$\mathcal{O}(n^2 m)$

The effect of those faults is the same that can be produced by a cyberattack targeting industrial devices. In fact, cyberattacks on these environments are different from traditional scenarios and they are very specialised attack and mainly consist in alter values in the industrial process as highlighted in ref. [5].

In the TEP testbed, four reactants in a gaseous state (A, D, E, and C) are injected into the input pipeline, and they are combined to generate two products (G and H) and one by-product (F). The whole simulation testbed is shown in Figure 1. The testbed comprises five major modules: reactor, condenser, liquid-vapour separator, compressor, and stripper. The four aforementioned reactants are fed into the reactor generating products in the form of vapour and unreacted components. Then, the vapour components are converted into a liquid employing a cooler located in the condenser. Next, liquid and unreacted components are injected into the liquid-gas separator, where the unreacted components are recycled and reinjected at the input pipeline. In contrast, the liquid components are passed to a product stripping unit where the remaining reactants are filtrated to generate products G and H.

The authors of the testbed generated a dataset and made it publicly available. However, we used the dataset generated in 2017 by Rieth et al. [39] because it is a much richer dataset with several simulations per anomaly type. In particular, Rieth et al. generated four files containing training and test datasets with 20 types of anomalies and training and test datasets free of anomalies. Each file consists of 500 simulations for each anomaly type. The training files contain 500 samples in each

simulation, while the test files have 960 samples in each simulation. Each sample consists of 52 features, including 41 measurement variables and 11 manipulated variables sampled every 3 min, giving 25 h for the training datasets and 48 h for the test datasets. In addition, the authors included a column called *faultNumber* that indicates the type of anomaly. A value of 0 means that the sample is normal, and values from 1 to 20 mean that the sample belongs to one of the anomalous classes.

5.2 | Dataset preprocessing

First, we split the dataset provided by the authors into training, validation, and test datasets. The authors provide the dataset in four different files: faulty training, free-fault training, faulty testing, and free-fault testing. We used the free-fault training file as our training and validation dataset. Since there are 500 simulations with 500 samples each, we selected the training dataset as the first 350 samples of even simulations and the last 250 samples of odd simulations. The validation dataset comprised the remaining samples: the last 150 samples of even simulations and the first 250 samples of odd simulations. Finally, the union of faulty testing and free-fault testing files was selected as our test dataset. The first 160 samples (8 h) of each simulation in the test file were removed since they were normal samples.

Next, we studied the features to discover spurious and corrupted values. In this case, we excluded anomalies 3, 9, and

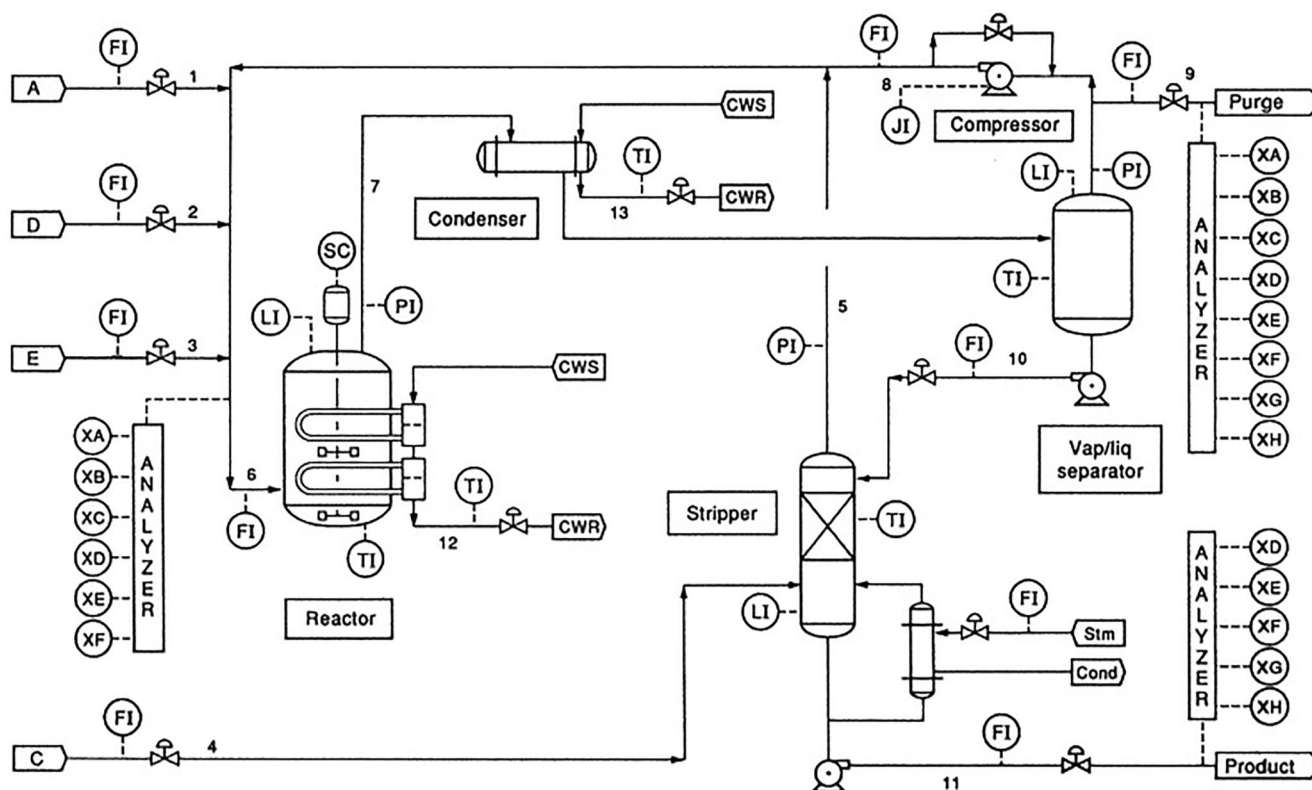


FIGURE 1 Tennessee Eastman process.

15 because the perturbation introduced was not sufficient to produce an anomaly [40]. Therefore, all samples labelled with fault 3, 9, or 15 were removed.

Finally, we computed the mean and standard deviation of feature values in the training dataset and used those metrics to scale the training, validation and test datasets.

5.3 | Feature filtering

This section shows the insight obtained after applying the techniques suggested by the second step of our proposal, which tries to remove those features that do not contribute to the model.

First, we performed a correlation study to discover if data leakage exists between the features and the label in the dataset. To compute the correlation, we used Pearson's correlation coefficient. In this specific case, nine pairs of features are correlated above the 90%, being XMV_7 and XMEAS_12 and XMV_8 and XMEAS_15 the most correlated features with 99.99%. However, we did not observe a high correlation between any of the features and the label (faultNumber in this case). In fact, the highest correlation that we found was 15.39% with XMEAS_22 feature. Because of the lack of high correlation between the label and features, we concluded that data leakage did not exist, and the problem is sufficiently

complex to apply advanced ML/DL techniques. Consequently, we did not remove any feature in light of these results.

Second, we studied the variance of the training dataset to find out features whose values did not change in the whole dataset. Figure 2 shows the 10 features with the lowest variance. In this case, we can observe that all features suffer a change in their values. Therefore, in this stage, we did not remove any feature.

Finally, we performed a study regarding the statistical distribution of features between the training, validation, and test datasets. This step is needed because ML and DL techniques are only applicable when the statistical distribution between the datasets is similar. The Kolmogorov–Smirnov (K-S) test is a good technique to measure the similarity between distributions, which is computed using the Empirical Cumulative Distribution Function (ECDF). The closer K-S statistic is to 0, the more likely it is that both features are drawn from the same distribution. In this specific case, we observed that the features from all datasets were generated from the same distribution. In fact, the highest K-S statistic found was 0.061, and it was reported by the XMV_9 feature, followed by the XMEAS_19 with a K-S statistic of 0.059. Figure 3 shows, for the feature with the highest K-S statistic, the empirical cumulative distribution from which the K-S test was computed, the K-S statistic itself, the histogram, and the value over time. As can be observed, the ECDF of the different datasets are similar. Furthermore, the histograms and the measures over time of all the datasets are also similar.

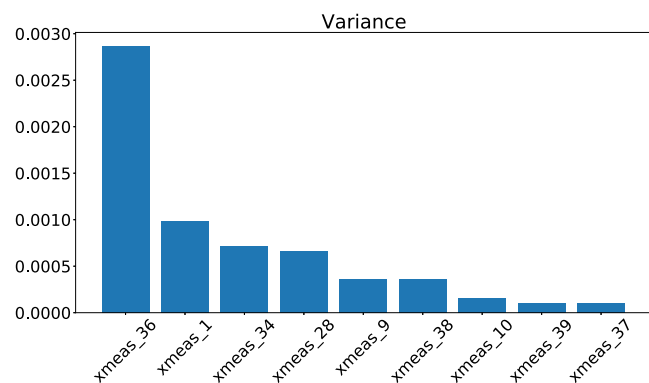


FIGURE 2 The 10 features with the lowest variance.

5.4 | Feature extraction

This step applied the autocorrelation and DFT techniques to extract higher-order features that help the model to discriminate between normal and abnormal behaviour. These techniques were applied over a window of data. The three dominant frequencies obtained from the DFT, together with the three dominant coefficients returned by the autocorrelation technique, were added to the dataset. The size input window of these techniques was a hyper-parameter that will be shown in the next section. After applying this step, the dataset contained

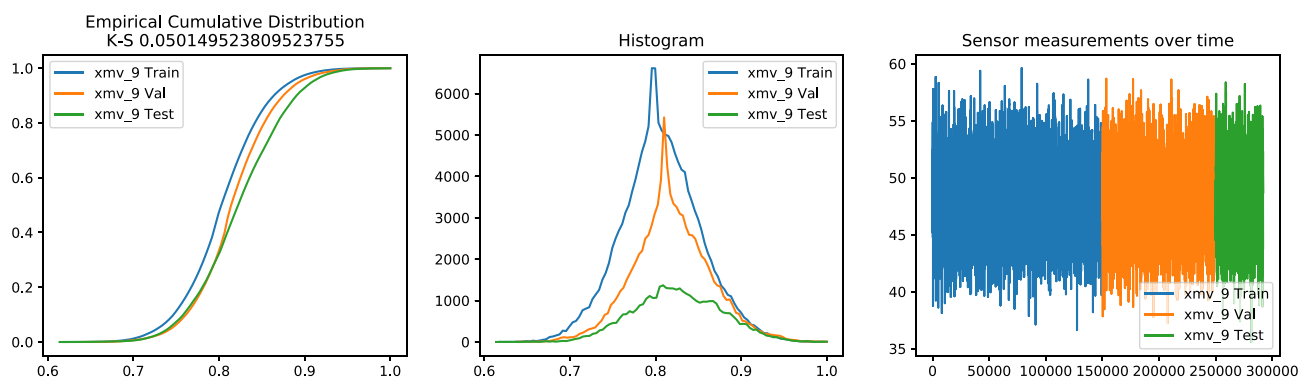


FIGURE 3 Empirical cumulative distributions, histograms, and measures over time of the two features with the highest K-S statistic for the training, validation and test datasets.

364 features. In particular, the original dataset contained 52 features, and each technique mentioned above (autocorrelation and DFT) generated 156 features (3 values for each of the 52 original features). Finally, we computed the mean and standard deviation of these features from the training dataset to scale the training, validation, and test datasets since the new features were in different ranges.

5.5 | Anomaly detection method

First, we considered models dealing with time-series data, such as LSTM or 1D-CNN. In fact, we chose those two models and let the fine-tuning process decide which of both achieved the best performance, that is, the lowest reconstruction error over the validation dataset. The selected models received a sequence of sensor data as input and predicted the value for those sensors 6 minutes (two timesteps) in the future.

Second, we defined the hyperparameters and their range to be tested during the fine-tuning process. The hyperparameters and their range can be seen in Table 4.

Third, we performed a random search to determine the best hyperparameters combination. Strictly speaking, a grid-search strategy is the best approach. However, the large number of hyperparameters combinations we had would require a considerable amount of time to accomplish this task. We selected a random combination of hyperparameters and trained the model with the training dataset to perform the search. The number of epochs of each training was set to 1 500, and an early stopping procedure was configured to stop the training if the validation error did not decrease for 150 epochs. Once the training is finished, the mean squared error of the validation dataset is computed. The total number of random combinations was configured to be 50. The final

hyperparameters selected were those that achieved the lowest mean square error over the validation dataset.

Table 5 shows both the model architecture and its hyperparameters that achieved the lowest reconstruction error over the validation dataset. The model was trained with the Adam optimiser. The optimal learning rate was 0.001, and the optimal window sizes used with the model and feature extraction were 20 and 10 respectively.

Fourth, once the model was trained and fine-tuned, we used the validation dataset to compute the z-score. To do this, we computed the error between the predicted value of each sensor in the next timestep and its actual value in the training dataset. Then, we computed the mean and standard deviation of these errors, and they were used to scale the prediction errors in the validation dataset. Next, to ignore the potential outliers, we used Equation (5) to compute the final threshold. Finally, we selected the highest z-score value among all features, as explained in Section 4.

TABLE 5 Architecture of model that achieved the lowest reconstruction error over the validation dataset.

Layers	Hyperparameters
1D convolutional layer	Filters: 256
	Kernel size: 3
	Padding: Same
LeakyReLU	Alpha: 0.5
Max pooling	Pool size: 2
1D convolutional layer	Filters: 128
	Kernel size: 3
	Padding: Same
LeakyReLU	Alpha: 0.5
Max pooling	Pool size: 2
Dense layer	Neurons: 1024
Dropout	Rate: 0

TABLE 4 Set of hyperparameters tested.

Hyper-parameter	Values tested
Type	CNN, LSTM
Numbers of CNN layers	2, 3
Numbers of LSTM layers	2, 3
Number of dense layers	1, 2
Number of CNN neurons	128, 256, 512
Number of LSTM neurons	128, 256, 512
Number of dense neurons	512, 1024
CNN kernel size	3, 5
Dropout	0, 0.5, 0.7
Learning rate	0.001, 0.003
Activation function	ReLU, LeakyReLU
Sequence input length (feature extraction and DL model)	10, 20

Abbreviations: CNN, Convolutional Neural Networks; LSTM, Long Short-Term Memory.

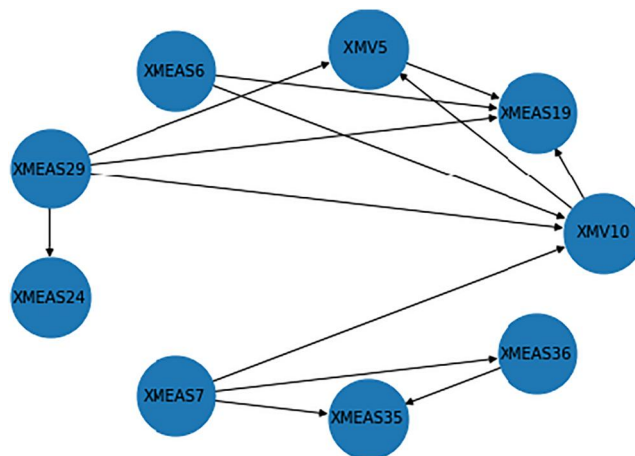


FIGURE 4 Graph built during anomaly 4.

5.6 | Interpretability

In this step, we grouped the original, autocorrelation, and frequency features by the sensor/actuator from which they were obtained. In this way, we grouped the features in 52 sets containing seven features each. Each set represents one particular sensor/actuator being monitored and contains one feature from the original dataset, together with three autocorrelation features and three frequency features extracted from the original feature. Then, we established that when any of these six features exceeded the threshold, the anomaly affected that sensor/actuator.

Additionally, we computed the dCor values between each pairs of feature in the validation dataset. This served to construct the adjacency matrix to build the relationship graph in further steps. To speed up this process, we only used 1000 samples to compute the dCor for each pair of features.

Then, when an anomaly was detected, we created a batch of samples with a size large enough to contain representative data of the relationship between features during this anomaly, for example, 2000. To prevent large computation time, we computed

the 10 features that exceeded the threshold more frequently. In case that more than 10 features exceeded the threshold with the same frequency, all these features were considered. Next, to build the relationship graph, we only considered those pairs of features whose dCor value exceeded their corresponding validation dCor. This process is intended to remove the underlying similarity of the normal behaviour. Finally, we used IGCI for each pair of features to determine which of them is the cause and which is the effect. An example of these cause/effects graph can be seen in Figure 4, where the relationship between features during the anomaly 15 is showed. In this example, we can see that the nodes of XMEAS_6, XMEAS_7, and XMEAS_29 are presumably the causes of the anomaly 4.

5.7 | Validation

First, we defined the metrics to measure the detection performance of our implementation. We used Precision, Recall, and F1-score, which are the common metrics to measure the detection performance in AD.

TABLE 6 Recall rates for the different works.

#	FCNN [20]	DBN [14]	GAN-DBN [14]	GAN-SRCC-DBN [14]	GAN-PCC-DBN [14]	GAN-MI-DBN [14]	FCNN [21]	IPCA [15]	Ours
0	0.21	0.941	0.989	0.997	0.955	0.994	-	-	0.9575
1	0.98	0.978	0.997	0.997	0.996	0.993	1	1	0.9917
2	0.99	0.951	0.981	0.998	0.984	0.985	0.9951	0.9954	1
3	0.25	0.208	0.241	0.330	0.402	0.283	-	-	-
4	0.98	0.972	0.983	0.992	0.988	0.977	1	1	1
5	0.98	0.936	0.938	0.967	0.974	0.971	1	1	1
6	1	0.992	0.999	0.999	0.997	0.991	1	1	1
7	1	0.957	0.971	0.973	0.981	0.971	1	1	1
8	0.98	0.934	0.943	0.950	0.954	0.935	0.9806	0.9846	0.9990
9	0.27	0.191	0.201	0.422	0.410	0.280	-	-	-
10	0.89	0.946	0.962	0.965	0.974	0.961	0.9396	0.9157	0.9542
11	0.94	0.948	0.948	0.971	0.955	0.970	0.9720	0.9567	0.9838
12	0.99	0.965	0.971	0.993	0.990	0.994	0.9869	0.9978	0.9997
13	0.96	0.917	0.931	0.957	0.940	0.955	0.9578	0.9623	0.8982
14	0.98	0.927	0.934	0.944	0.948	0.920	0.9997	1	1
15	0.31	0.471	0.510	0.633	0.552	0.603	-	-	-
16	0.92	0.775	0.907	0.924	0.919	0.902	0.9541	0.9570	0.9926
17	0.98	0.780	0.822	0.935	0.901	0.926	0.9593	0.9684	0.9032
18	0.98	0.914	0.943	0.955	0.961	0.949	0.9415	0.9515	0.8107
19	0.90	0.945	0.959	0.982	0.970	0.981	0.9918	0.9917	0.8669
20	0.91	0.881	0.900	0.952	0.932	0.9600	0.9362	0.9650	0.8804
Overall	0.8285	0.8347	0.8586	0.8970	0.8915	0.8810	0.9773	0.9792	0.9577

Note: Bold values indicate which approach achieved the best performance.

Abbreviations: DBN, Deep Belief Network; FCNN, Fully Connected Neural Networks; GAN, Generative Adversarial Network; IPCA, Interval PCA.

To compute these metrics, and considering that our approach is semi-supervised, we converted the labels of the TEP dataset, that is, the faultNumber column, to binary, being 0 a normal sample and 1 an abnormal sample. Then, we evaluated our system with the test dataset. When any of the features of a sample exceeded the threshold previously computed, we predicted that sample as class 1 (abnormal). Otherwise, the sample was predicted as class 0 (normal).

Second, we compared our work with the literature. In particular, Table 6 shows the recall rates per anomaly type achieved by different works. It is worth mentioning that we only have compared with those works that offer a recall rate and break down the results by anomaly type. For this reason, the solutions presented in refs. [17, 26], and [18] are not considered in this section.

As can be seen, our system achieved the best result in the majority of anomaly types. To be specific, we achieved the best results in 10 of them (2, 4, 5, 6, 7, 8, 11, 12, 14, and 16). In contrast, the solutions presented in refs. [15, 21] are the second and third that achieved the highest number of best results with the anomaly types 8 and 6 respectively. Finally, the solutions presented in [14, 20] achieved the best result in the anomaly types 4 and 2 respectively.

In terms of the general result, the solution in ref. [15] achieved the best result with a recall of 0.9792, followed by the solution proposed in ref. [21] that achieved a recall of 0.9773.

Our solution is located in the third position achieving a recall of 0.9577. This may be due to the anomalies 17, 18, 19, and 20, where our system achieved the lowest results.

Finally, considering all anomalies, our system presents an excellent precision rate with a result of 0.9974, a recall rate above the average of the other works, as we saw previously, with a result of 0.9577, and an F1-score of 0.9771.

However, most of the related works were developed following a supervised approach or, at least, a supervised approach was used during the training of one part of the models. To be specific, only our system and the solution proposed in ref. [15] followed a purely semi-supervised approach.

Additionally, none of the works in the literature focussed on AD provides information about the sensors or actuators where the anomaly is produced. On the one hand, supervised approaches only indicate whether an anomaly exists. In the best case, they can inform about the probability of each type of anomaly to occur. On the other hand, IPCA, which is a semi-supervised approach, is based on PCA that performs a dimensionality reduction process, thus preventing the identification of the point where the anomaly is produced. Besides, solutions focussed on fault diagnosis such as refs. [17, 18] do offer information about where the fault is produced. However, those solution have a significant limitation. Specially, both solutions selected a subset of features to develop their

TABLE 7 Anomalies with its description and the potential causes.

#	Anomaly description	Causes of anomaly	Feature description
1	A/C feed ratio, B composition constant	XMEAS_9	Reactor temperature
2	B composition, A/C ration constant	XMEAS_30	Component B (purge gas analysis)
4	Reactor cooling water inlet temperature	XMEAS_6	Reactor feed rate
		XMEAS_7	Reactor pressure
		XMEAS_29	Component A (purge gas analysis)
5	Condenser cooling water inlet temperature	XMEAS_30	Component B (purge gas analysis)
6	A feed loss	XMV_3	A feed flow
7	C Header pressure loss-reduced availability	XMEAS_41	Component H (product analysis)
8	A, B, and C feed composition	XMEAS_21	Reactor water temperature
10	C Feed temperature	XMEAS_7	Reactor pressure
		XMEAS_13	Separator pressure
		XMEAS_16	Stripper pressure
		XMEAS_41	Component H (product analysis)
11	Reactor cooling water inlet temperature	XMEAS_9	Reactor temperature
		XMEAS_29	Component A (purge gas analysis)
12	Condenser cooling water inlet temperature	XMEAS_16	Stripper pressure
13	Reactor kinetics	XMEAS_20	Compressor work
		XMEAS_21	Reactor water temperature
14	Reactor cooling water valve	XMEAS_9	Reactor temperature
		XMEAS_41	Component H (product analysis)

approaches, invalidating their results in terms of determining causes of faults.

In contrast, our system is designed with interpretability in mind and considering all the sensors/actuators in the industrial setting. Therefore, it can determine the actuator/sensor affected by the anomaly and their causes and effects.

Table 7 shows the features that presumably caused the anomalies. The anomaly 15–20 were ignored because they are not documented and, therefore, we cannot link the root of the anomaly with the features. In contrast, anomalies 1–15 were considered, and for each one, Table 7 shows its anomaly number, description, features that caused the anomaly, and its description.

6 | CONCLUSIONS AND FUTURE WORK

This work presented an interpretable and semi-supervised system to detect cyberattacks on industrial control systems. The core of our proposal is the introduction of a new step focussed on interpretability that uses dCor and IGCi to determine the effect of each anomaly. Furthermore, we also present specific steps to carry out both the feature extraction and AD model selection. In addition, we validated our proposal using the TEP testbed, which is a simulation testbed frequently used in AD. Our system achieved state-of-the-art in most anomaly types and the third-best overall recall rate. To be specific, it reached the best result in 11 out of a total of 20 anomaly types and achieved an overall recall rate of 0.9577.

As future work, we contemplate to study the anomalies in which our system achieved a low recall rate in order to improve the results. Moreover, we intend to test our modification in other real testbeds, and we also consider to make a study of origin of the outliers filtered with the IQR technique. Our hypothesis is that once we know the origin, we can reduce the number of outliers. Additionally, we plan to test other interpretability methods to explain the adversarial samples generated in industrial scenarios.

AUTHOR CONTRIBUTIONS

Ángel Luis Perales Gómez: Conceptualisation; data curation; formal analysis; investigation; methodology; software; writing—original draft. **Lorenzo Fernández Maimó:** Data curation; formal analysis; investigation; methodology; software; writing—original draft. **Alberto Huertas Celdrán:** Data curation; formal analysis; investigation; supervision; validation; writing—review & editing. **Félix J. García Clemente:** Conceptualisation; funding acquisition; investigation; project administration; supervision; validation; writing—review & editing.

ACKNOWLEDGEMENTS

This work has been funded under grant TED2021-129300B-I00, by MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, UE, grant PID2021-122466OB-I00, by MCIN/AEI/10.13039/501100011033/FEDER, UE, and by

the Swiss Federal Office for Defence Procurement (armasuisse) (project code Aramis R-3210/047-31).

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ORCID

Ángel Luis Perales Gómez  <https://orcid.org/0000-0003-1004-881X>

REFERENCES

- Xu, LD., Duan, L.: Big data for cyber physical systems in industry 4.0: a survey. *Enterprise Inf. Syst.* 13(2), 148–169 (2019). <https://doi.org/10.1080/17517575.2018.1442934>
- Lee, J., et al.: Industrial artificial intelligence for industry 4.0-based manufacturing systems. *Manufact. Lett.* 18, 20–23 (2018). <https://doi.org/10.1016/j.mfglet.2018.09.002>
- Ani, U.P.D., He, H., Tiwari, A.: Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective. *J. Cyber Secur. Technol.* 1(1), 32–74 (2017). <https://doi.org/10.1080/23742917.2016.1252211>
- Hemsley, K.E., Fisher, E.: History of Industrial Control System Cyber Incidents. Idaho National Lab (INL), Idaho Falls, ID (United States) (2018)
- Perales Gómez, Á.L., et al.: On the generation of anomaly detection datasets in industrial control systems. *IEEE Access* 7, 177460–177473 (2019). <https://doi.org/10.1109/access.2019.2958284>
- Downs, J.J., Vogel, E.F.: A plant-wide industrial process control problem. *Comput. Chem. Eng.* 17(3), 245–255 (1993). [https://doi.org/10.1016/0098-1354\(93\)80018-i](https://doi.org/10.1016/0098-1354(93)80018-i)
- Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv.* 41(3), 1–58 (2009). <https://doi.org/10.1145/1541880.1541882>
- Zhou, X., et al.: Variational LSTM enhanced anomaly detection for industrial big data. *IEEE Trans. Ind. Inf.* 17(5), 3469–3477 (2020). <https://doi.org/10.1109/tii.2020.3022432>
- Zhou, X., et al.: Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems. *IEEE Trans. Ind. Inf.* 17(8), 5790–5798 (2020). <https://doi.org/10.1109/tii.2020.3047675>
- Huma, Z.E., et al.: A hybrid deep random neural network for cyberattack detection in the industrial internet of things. *IEEE Access* 9, 55595–55605 (2021). <https://doi.org/10.1109/access.2021.3071766>
- Khan, I.A., et al.: Enhancing IIoT networks protection: a robust security model for attack detection in Internet Industrial Control Systems. *Ad. Hoc. Netw.* 134, 102930 (2022). <https://doi.org/10.1016/j.adhoc.2022.102930>
- Al-Abassi, A., et al.: An ensemble deep learning-based cyber-attack detection in industrial control system. *IEEE Access* 8, 83965–83973 (2020). <https://doi.org/10.1109/access.2020.2992249>
- Boateng, E.A., Bruce, J., Talbert, D.A.: Anomaly detection for a water treatment system based on one-class neural network. *IEEE Access* 10, 115179–115191 (2022). <https://doi.org/10.1109/access.2022.3218624>
- Tian, W., et al.: Identification of abnormal conditions in high-dimensional chemical process based on feature selection and deep learning. *Chin. J. Chem. Eng.* 28(7), 1875–1883 (2020). <https://doi.org/10.1016/j.cjche.2020.05.003>
- Basha, N., et al.: Multiclass data classification using fault detection-based techniques. *Comput. Chem. Eng.* 136, 106786 (2020). <https://doi.org/10.1016/j.compchemeng.2020.106786>
- Wang, Z., et al.: Explaining the attributes of a deep learning based intrusion detection system for industrial control networks. *Sensors* 20(14), 3817 (2020). <https://doi.org/10.3390/s20143817>

17. Ziaei-Halimejani, H., Zarghami, R., Mostoufi, N.: Joint recurrence based root cause analysis of nonlinear multivariate chemical processes. *J. Process Control* 103, 19–33 (2021). <https://doi.org/10.1016/j.jprocont.2021.05.008>
18. Ji, C., et al.: Real-time industrial process fault diagnosis based on time delayed mutual information analysis. *Processes* 9(6), 1027 (2021). <https://doi.org/10.3390/pr9061027>
19. Selim, G.E.I., et al.: Anomaly events classification and detection system in critical industrial internet of things infrastructure using machine learning algorithms. *Multimed. Tool. Appl.* 80(8), 12619–12640 (2021). <https://doi.org/10.1007/s11042-020-10354-1>
20. Sokolov, A.N., Pyatnitsky, I.A., Alabugin, S.K.: Applying methods of machine learning in the task of intrusion detection based on the analysis of industrial process state and ICS networking. *FME Trans.* 47(4), 782–789 (2019). <https://doi.org/10.5937/fmet1904782s>
21. Heo, S., Lee, J.H.: Fault detection and classification using artificial neural networks. *IFAC-PapersOnLine* 51(18), 470–475 (2018). <https://doi.org/10.1016/j.ifacol.2018.09.380>
22. Khan, I.U., et al.: A proactive attack detection for heating, ventilation, and air conditioning (HVAC) system using explainable Extreme gradient boosting model (XGBoost). *Sensors* 22(23), 9235 (2022). <https://doi.org/10.3390/s22239235>
23. Munir, M., et al.: Pattern-based contextual anomaly detection in HVAC systems. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1066–1073. IEEE (2017)
24. Perales Gómez, Á.L., et al.: SafeMan: a unified framework to manage cybersecurity and safety in manufacturing industry. *Software Pract. Ex.* 51(3), 607–627 (2021). <https://doi.org/10.1002/spe.2879>
25. Perales Gómez, Á.L., et al.: MADICS: a methodology for anomaly detection in industrial control systems. *Symmetry* 12(10), 1583 (2020). <https://doi.org/10.3390/sym12101583>
26. Morales-Forero, A., Bassetto, S.: Case study: a semi-supervised methodology for anomaly detection and diagnosis. In: 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 1031–1037. IEEE (2019)
27. Miller, T.: Explanation in artificial intelligence: insights from the social sciences. *Artif. Intell.* 267, 1–38 (2019). <https://doi.org/10.1016/j.artint.2018.07.007>
28. Robnik-Šikonja, M., Bohanec, M.: Perturbation-based explanations of prediction models. In: *Human and Machine Learning*, pp. 159–175. Springer (2018)
29. Rousseeuw, P.J., Hubert, M.: Robust statistics for outlier detection. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 1(1), 73–79 (2011). <https://doi.org/10.1002/widm.2>
30. Székely, G.J., Rizzo, M.L., Bakirov, N.K.: Measuring and testing dependence by correlation of distances. *Ann. Stat.* 35(6), 2769–2794 (2007). <https://doi.org/10.1214/009053607000000505>
31. Huo, X., Székely, G.J.: Fast computing for distance covariance. *Technometrics* 58(4), 435–447 (2016). <https://doi.org/10.1080/00401706.2015.1054435>
32. Shao, F., Liu, H.: The theoretical and experimental analysis of the maximal information coefficient approximate algorithm. *J. Syst. Sci. Inf.* 9(1), 95–104 (2021). <https://doi.org/10.21078/jssi-2021-095-10>
33. Heller, R., Heller, Y., Gorfine, M.: A consistent multivariate test of association based on ranks of distances. *Biometrika* 100(2), 503–510 (2013). <https://doi.org/10.1093/biomet/ass070>
34. Daniusis, P., et al.: Inferring deterministic causal relations. In: 26th Annual Conference on Uncertainty in Artificial Intelligence (2010)
35. Chollet, F.: Keras. GitHub (2015). <https://github.com/fchollet/keras>
36. Virtanen, P., et al.: SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* 17, 261–272 (2020)
37. Kalainathan, D., Goudet, O.: Causal discovery toolbox: uncover causal relationships in python preprint arXiv:190302278. arXiv (2019)
38. Vallat, R.: Pingouin: statistics in Python. *J. Open Source Softw.* 3(31), 1026 (2018). <https://doi.org/10.21105/joss.01026>
39. Rieth, C.A., et al.: Issues and advances in anomaly detection evaluation for Joint human-automated systems. In: *International Conference on Applied Human Factors and Ergonomics*, pp. 52–63. Springer (2017)
40. Onel, M., Kieslich, C.A., Pistikopoulos, E.N.: A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the Tennessee Eastman process. *AIChE J.* 65(3), 992–1005 (2019). <https://doi.org/10.1002/aic.16497>

How to cite this article: Perales Gómez, Á.L., et al.: An interpretable semi-supervised system for detecting cyberattacks using anomaly detection in industrial scenarios. *IET Inf. Secur.* 17(4), 553–566 (2023). <https://doi.org/10.1049/ise2.12115>